

第3章 RFR スワップとマルチカーブ

この章では初めに TONA や SOFR 等を変動金利とする RFR カーブ/RFR スワップについて説明する。RFR カーブは Ibor 廃止後のベンチマークイールドカーブとして重要であり、正確に理解する必要がある。

次に RFR カーブをディスカウントカーブとするマルチカーブの計算法とそのコードを紹介する。

3.1 オーバーナイト インデックス スワップ入門

第2章では6ヶ月 Tibor が変動金利であった金利スワップを説明した。ここでは無担保コール翌日物金利 **TONA**(Tokyo OverNight Average rate) を変動金利とするオーバーナイトインデックススワップ(以下 OIS)について考察する。

通貨により使用される翌日物金利は異なるが、例えば USD では **SOFR** (Secured Overnight Financing Rate: 担保付翌日物調達金利)、EURO では **ESTR**(Euro Short-Term Rate: ユーロ短期金利) が使用される。TONA を含め、これらの翌日物金利は **RFR**(Risk Free Rate) として参照される。

6ヶ月 Tibor のスワップでは、指標変動金利は**利息期間** (accrual period) の期首に決定され(前決め金利, forward looking rate)、6ヶ月後に固定金利と交換した。

OIS では毎日の市場引け後にその日の RFR が決定されるが、金利の交換は日次ベースでは行われない。OIS の変動レグの金利は、利息期間中に発表された日々の RFR を複利で計算したレート(後決め金利, backward looking rate, setting-in-arrears rate)となるのが一般的である。この計算を**デイリーコンパウンド**と呼び、3.1.4 節で具体的な計算を例示する。

OIS の標準的な利払頻度は年1回である。スワップ期間が1年以内の場合、スワップ満期日の2営業日後に一度のみ金利を交換する。この“2営業日後”は **Two-days Lag**(以下 2日ラグ) と呼ばれる。スワップ期間が1年超では、(2日ラグを伴って)1年毎に金利交換が行われる。

これらの点を除けば、OIS は第2章で説明した6ヶ月 Tibor の金利スワップ

プと同じ計算が行われる。従ってこれまで使用した計算式^{*1} はほぼそのまま利用可能である。

3.1.1 TONA カーブ

初めに図 3.1 で TONA スワップレートからディスカウントファクターを算出する例を示す。その計算過程から Tona スワップカーブ(以下 TONA カーブ)^{*2} のイメージを作ろう。図 3.1 は次の 3 つのセルから構成。

1 番セル : Tona スワップレートのデータ。(1 番セルの上にオブジェクト図を描いた)

2 番セル : makeTonaCurve 関数を定義。(2.2.4 節 makeTiborCurve の Tona 版)

3 番セル : TONA カーブに関して、以下の 3 つを算出。

- ディスカウントファクター (tonaDF 列)
- パーレート (parRT 列)
- ゼロレート (zeroRT 列)
- ゼロレートを出力させた理由は 3.1.2 節でカーブをシフトする為で、TONA カーブの本質からはズレる。

まず、出力結果で 8 月 22 日を除き、parRT 列と zeroRT 列が異なっている点に注意しよう。(TONA スワップの 1 年までのレートは満期時のみのキャッシュフローで、ゼロレートに類似。従って、本来 parRT=zeroRT となるべき。)

この理由は parRT 列がフォワードレートの為となるが、この問題意識を持って、次の TONA カーブの特徴を理解しよう。

取引日 : 3 番セル 1 行目で取引日 (tradeDT) を 2022 年 8 月 19 日に設定。

TONA をノードに持つカーブ : 1 番セル crvDATA の 1 つ目の要素 ('depo', '1d', -0.009) は当日決済の TONA レート。(このレートの満期日は翌営業日の 8 月 22 日)

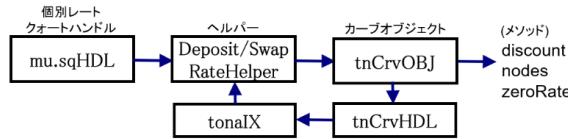
イールドカーブの決済日 : この当日決済の TONA がイールドカーブに入っている為、カーブ決済日は取引日の当日。(T+0)

- ディスカウントファクターが 1.0 となる日も取引日当日。(T+0)

TONA スワップレート : 1 番セル crvDATA 変数の 2 つ目以降は TONA スワップレートで T+2 決済。

^{*1}OIS の計算式の導出は Lyashenko and Mercurio [41] を参照。

^{*2}同様に RFR を変動金利とするスワップカーブを **RFR カーブ** と参照。



```

1 from myABBR import * ; import myUtil as mu
2 crvDATA = [ ('depo', '1d', -0.009), ('swap', '1w', -0.01261), ('swap', '2w', -0.01469),
3             ('swap', '1m', -0.01807), ('swap', '3m', -0.01919), ('swap', '6m', -0.01043),
4             ('swap', '9m', 0.00022), ('swap', '12m', 0.0125), ('swap', '18m', 0.03125),
5             ('swap', '2y', 0.04875), ('swap', '3y', 0.07375), ('swap', '4y', 0.09479),
6             ('swap', '5y', 0.11854), ('swap', '7y', 0.19146) ]

```

```

1 def makeTonaCurve(crvDATA):
2
3     # 1. 指標金利オブジェクト
4     tnCrvHDL = ql.RelinkableYieldTermStructureHandle()
5     tonalIX = ql.OVERNIGHTIndex('TONA', Tp0, jpyFX, calJP, dcA365, tnCrvHDL)
6
7     # 2. カーブヘルパー
8     cHelper, tnParRATE = [], []
9     for knd, tnr, rt in crvDATA:
10         if knd == 'depo':
11             cHelper.append(ql.DepositRateHelper(mu.sqHDL(rt/100), tonalIX))
12         if knd == 'swap':
13             cHelper.append(ql.OISRateHelper(Tp2, pD(tnr), mu.sqHDL(rt/100), tonalIX))
14             tnParRATE.append(rt/100) # パーレート用リスト
15
16     # カーブオブジェクト
17     tnCrvOBJ = ql.PiecewiseLogLinearDiscount(Tp0, calJP, cHelper, dcA365)
18     tnCrvHDL.linkTo(tnCrvOBJ); tnCrvOBJ.enableExtrapolation()
19     return [tonalIX, tnCrvOBJ, tnCrvHDL, tnParRATE] # 4つのオブジェクトを戻す

```

```

1 tradeDT = jDT(2022, 8, 19) ; setEvDT(tradeDT)
2 tonalIX, tnCrvOBJ, tnCrvHDL, tnParRATE = makeTonaCurve(crvDATA)
3 dfTONA = pd.DataFrame([(dt, df, pr, tnCrvOBJ.zeroRate(dt, dcA365, cmpdSPL, freqA).rate())
4                         for (dt, df), pr in zip(tnCrvOBJ.nodes()[1:], tnParRATE)],
5                         columns=['date', 'tonaDF', 'parRT', 'zeroRT'])
6 print("決済日(reference): ", tnCrvOBJ.referenceDate().ISO())
7 fmtSCF.update(tonaDF='[:11f]')
8 display(dfTONA[:4].style.format(fmtSCF))

```

決済日(reference): 2022-08-19

| | date | tonaDF | parRT | zeroRT |
|---|----------------------|---------------|------------|------------|
| 0 | August 22nd, 2022 | 1.00000073973 | -0.009000% | -0.009000% |
| 1 | August 30th, 2022 | 1.00000350357 | -0.012610% | -0.011625% |
| 2 | September 6th, 2022 | 1.00000671977 | -0.014690% | -0.013626% |
| 3 | September 26th, 2022 | 1.00001791784 | -0.018070% | -0.017210% |

図 3.1: TONA スワップカーブ

フォワードレート: 従って TONA スワップレート (以下 OIS レートとしても参照) は 2 日先スタートのフォワードレートである。

- 例えば 1w (1 週間) の OIS レート -0.01261% は 2 日先スタート (8 月 23 日) のテナー 1 週間 (8 月 30 日満期) のレート。

ディスカウントファクター計算式: S 年先スタート満期 E 年のフォワードレート $f_{S \times E}$ のディスカウントファクター D_E は式 (2.6) で D_E について

て解いた次式より計算。

$$D_E = D_S \times \left(\frac{1}{1 + f_{S \times E} \times \tau} \right), \quad \tau : E - S \quad (3.1)$$

- しかし、8月23日と30日のディスカウントファクターを $D_{8/23}$ 、 $D_{8/30}$ と表すと、 $D_S = D_{8/23}$ 、 $D_E = D_{8/30}$ の2つの変数はいずれも未知数。(ただし、 $D_{8/22}$ は TONA レートから計算可能)
- そこで対数線形補間の次の式(式(1.3)参照)で連立方程式を作り、 D_S と D_E を算出する。

$$\frac{\ln(D_S) - \ln(D_{8/22})}{\ln(D_E) - \ln(D_{8/22})} = \frac{8/22 \sim 8/23 \text{ の日数}}{8/22 \sim 8/30 \text{ の日数}} = \frac{1}{8}$$

- レートを計算する式と補間法の式で連立方程式を作り、ディスカウントファクターを求める計算は 3.4.1 節で説明する**ブートストラップ法**と同じ。(スワップの一般的知識として理解しよう)
- 連立方程式を解くことは読者に任せ、ここではカーブオブジェクトの discount メソッドで計算した8月23日のディスカウントファクター 1.000001085 を使って計算を確認する。

```
1 tnCrv0BJ.discount(jDT(2022, 8, 23))
```

```
1.0000010852067613
```

- 式(3.1)により TONA スワップレート 1w(1週間)のディスカウントファクターは次式となり、図3.1の出力結果と同じ値となる。

$$\begin{aligned} D_{2022/8/30} &= \underbrace{1.000001085}_{D_{2022/8/23}} \times \frac{1}{\underbrace{1 - 0.01261\%}_{1w \text{ レート}} \times \frac{7 \text{ 日}}{365 \text{ 日}}} \\ &= 1.000003503 \end{aligned} \quad (3.2)$$

2日ラグ：この計算からも判るように TONA カーブのディスカウントファクター(以下 DF)や金利の計算に関して、2日ラグを考慮する必要は無い。

- ただし キャッシュフローの現在価値を計算する場合 2日ラグを考慮したキャッシュフロー日の DF が使用される。

以上が TONA カーブの特徴となる。では図3.1のコード等を確認しよう。

(makeTonaCurve 関数)

2番セルの makeTonaCurve 関数は図2.3と同じようなオブジェクト構成となり、以下では相違する部分のみを説明する。(図2.3のコードに不慣れな読者は比較しながら理解しよう)

4, 5: OvernightIndex : このコンストラクタによって、指標金利 TONA のオブジェクト tonaIX を作成。

QLP ドキュメントでは下図のようにコンストラクタの引数は最低 5 個。

```
ql.OvernightIndex(name, fixingDays, currency, calendar, dayCounter,
=ql.YieldTermStructureHandle())
```

```
name = 'CNYRepo7D'
fixingDays = 1
currency = ql.CNYCurrency()
calendar = ql.China()
dayCounter = ql.Actual365Fixed()
overnight_index = ql.OvernightIndex(name, fixingDays, currency, calendar, dayCounter)
```

- 2 番目の引数 fixingDays は金利が確定してから 決済日までの日数を指定。(1.3.2 節参照)
 - ◊ TONA は市場引け後に発表される為、T+0 を意味する **Tp0**(A.1 節参照) を指定。
- 6 番目の引数 YieldTermStructureHandle は Tibor コンストラクタ(図 1.14) と同様、変動レグの指標となる場合に省略は不可。

コード 5 行目の TONA オブジェクト tonaIX が指標金利となる為、4 行目で作成した空の tnCrvHDL(tona curve handle の略) を設定。

9, 10: depo : DepositRateHelper は図 1.13 を、mu.sqHDL 関数は 1.4.2 節を参照。

11, 12: OISRateHelper : このコンストラクタは OIS レートを持つ各ノードの情報を Helper オブジェクトに変換。

```
ql.OISRateHelper(settlementDays, tenor, fixedRate, overnightIndex,
discountingCurve=ql.YieldTermStructureHandle(), telescopicValueDates=False, paymentLag=0,
paymentConvention=ql.Following, paymentFrequency=ql.Annual, paymentCalendar=ql.Calendar(),
forwardStart=ql.Period(), overnightSpread=0.0, pillar=ql.Pillar.LastRelevantDate, customPillarDate=ql.Date())
```

```
forward6mLevel = 0.025
forward6mQuote = ql.QuoteHandle(ql.SimpleQuote(forward6mLevel))
yts6m = ql.FlatForward(0, ql.TARGET(), forward6mQuote, ql.Actual365Fixed() )
yts6mh = ql.YieldTermStructureHandle(yts6m)
oishelper = ql.OISRateHelper(2,ql.Period("3M"), ql.QuoteHandle(ql.SimpleQuote(0.01)), ql.Eonia(yts6mh),
,yts6mh, True)
```

図 3.2: OISRateHelper コンストラクタ

- OISRateHelper は上図のように最低 4 つの引数 settlementDays, tenor, fixedRate, overnightIndex が必要。
- 12 行目のコードで各引数を確認すれば、引数の意味は明らかな為説明は省略。

- 尚 2 番目の引数 **pD(tnr)** のように見慣れない小文字大文字(キャメルケース)の変数/関数が現れた場合 myABBR モジュールで定義された短縮形と考えよう。

13: tnParRate : TONA カーブは金利のベンチマークとして利用されるが、その際 パーレートとの単純な差を計算することが多い。(例えば 4.4 節の I スプレッド等)

- この tnParRate 変数はその為に用意。
- tnParRATE 変数は 1 番セルに記述された各レートと同じ値のリストとなる。(出力されたデータフレーム parRT 列を確認) *3

以上が makeTonaCurve 関数のコード部分へのコメントとなる。TONA カーブはベンチマーク以外にも割引カーブとしても使用される為、makeTonaCurve 関数を myUtil モジュールに組み入れ、再利用する。

尚、OISRateHelper コンストラクタ(図 3.2) は OIS を理解する際の重要な引数を持っており、その初期値に関して、補足しておく。

paymentLag=0 :

OIS カーブを算出する場合、7 番目の引数 paymentLag は 2 ではなく初期値の 0 を指定。

- 理論的に 2 日ラグの評価は Ibor 先物等で現れるコンベクシティー調整 (Convexity Adjustment) の 1 つの Timing Adjustment で行う。^{*4}
 - この理論の詳細は Henrard[31] 6 章を参照。そこでは“2 日ラグの Timing Adjustment は非常に小さい為 実務的には無視する”と記述。
- 式 (3.2) のようにフォワードレートやスワップレート等を計算する場合、2 日ラグは無視される。(具体的な数値例は図 3.5 で説明)
- 一方、キャッシュフローをディスカウントする場合、2 日ラグを考慮。(ラグのあるキャッシュフロー日のディスカウントファクターでキャッシュフローが割り引かれる。図 3.5 参照)

paymentConvention=Following :

8 番目の引数は Following が初期値となっている。もし ここを ModifiedFollowing にするには、5 番～8 番の引数を “tnCrvHDL, False, 0, mFLLW” のように指定。(TONA スワップは ModifiedFollowing であり、この修正は必要)

^{*3} この変数を作った理由は 15 行目のイールドカーブオブジェクト tnCrvOBJ から元データであるペーレートを再現することが面倒な為。

^{*4} コンベクシティー調整は測度変換の知識が必要であり、本書では扱わない。

(TONA ディスカウントカーブ)

図 3.1 の 3 番セルも図 2.3 に似たコードとなるが、3.1.2 節の準備として、パレート (parRT 列) とゼロレート (zeroRT 列) も出力させた。コードについて説明しよう。

1: 決済日 (reference) : tradeDT(取引日) は 8 月 19 日。出力結果から判るようカーブ決済日 (referenceDate) も同日。

2: makeTonaCurve : crvDATA を makeTonaCurve 関数に与え、TONA 指標オブジェクト tonaIX、カーブオブジェクトとハンドル tnCrvOBJ、tnCrvHDL、パレート tnParRATE を作成。

3~5: dfTONA : 主に tnCrvOBJ の情報から作成されるデータフレーム。コードが多少込み入っているので、以下のように別けて考えよう。

4: for(dt,df), pr in zip(tnCrvOBJ.nodes()[1:],tnParRATE) : リスト内包表記の for 文であり、日付 dt, ディスカウントファクター df, パレート pr の 3 つでループを処理。

- イテレータは tnCrvOBJ.nodes()[1:] と tnParRATE の **zip** で構成。
- tnCrvOBJ.nodes()[1:] のスライスで決済日を排除。
- nodes メソッドからはタプルとして日付 dt とディスカウントファクター df を取得。

3: tnCrvOBJ.zeroRate(dt,dcA365,cmpdSPL,freqA).rate() : カーブオブジェクト tnCrvOBJ の zeroRate メソッド (図 1.10 参照) を使い、各ノード日 (dt) のゼロレートを表示。(2 番~4 番の引数は TONA の市場慣例である dcA365, cmpdSPL, freqA を指定)

7: fmtSCF.update(tonaDF='{:11f}') : myABBR モジュールの fmtSCF 変数に tonaDF='{:11f}' を追加 (または修正)。

8: style.format(fmtSCF) : 2.4.2 節参照。

3.1.2 カーブシフト

図 3.1 の TONA カーブは現実のレートを使用した。その結果 出力された tonaDF 列は小数 6 桁目辺りでやっとゼロでない数字が現れた。

この不便を避ける為に 以下ではイールドカーブをシフトさせて、より判りやすい数値例で説明したい。

その準備として この 3.1.2 節では QuantLib でカーブをシフトさせる方法を紹介する。(ここで説明するカーブシフトは 4.4.3 節 Z スプレッド算出や一般的な金利シナリオの作成等で必須のツールであり、ぜひ使いこなせるようになろう)

通常のスワップカーブには預金レートやスワップレート等、異なる種類の金利で構成される為、イールドカーブを平行にシフト(パラレルシフト)させることはかなり難しい。そこでカーブを構成する各ノードのゼロレートをパラレルにシフトさせる方法が一般的である。

```

1 # TONAカーブの5%シフト
2 tnSFTOBJ = ql.ZeroSpreadedTermStructure(
3     tnCrvHDL, mu.sqHDL(5.0/100), compdSPL, freqA, dcA365)
4 tnSFTHDL = ql.YieldTermStructureHandle(tnSFTOBJ)
5 tnSFTIX = ql.OVERNIGHTINDEX('TONA', Tp0, jpyFX, calJP, dcA365, tnSFTHDL)
6 dfSFT = pd.DataFrame([(tnSFTOBJ.zeroRate(dt, dcA365, compdSPL, freqA).rate(),
7     tnSFTOBJ.discount(dt)) for dt in dfTONA.date], columns=['shftRT', 'DF'])
8 dfTnSFT = pd.concat([dfTONA, dfSFT], axis=1)
9 display(dfTnSFT[:4].style.format(fmtSCF))
10 # シフトカーブの使用
11 tnCrvOBJ = tnSFTOBJ; tnCrvHDL = tnSFTHDL; tonaIX = tnSFTIX

```

| | date | tonaDF | parRT | zeroRT | shftRT | DF |
|---|----------------------|---------------|------------|------------|-----------|------------|
| 0 | August 22nd, 2022 | 1.00000073973 | -0.009000% | -0.009000% | 4.991000% | 0.99958995 |
| 1 | August 30th, 2022 | 1.00000350357 | -0.012610% | -0.011625% | 4.988375% | 0.99849891 |
| 2 | September 6th, 2022 | 1.00000671977 | -0.014690% | -0.013626% | 4.986374% | 0.99754700 |
| 3 | September 26th, 2022 | 1.00001791784 | -0.018070% | -0.017210% | 4.982790% | 0.99483921 |

図 3.3: TONA カーブの 5%シフト

図 3.3 は図 3.1 で作成した TONA カーブをゼロレートベースで 5%シフトさせるコード例である。

図 3.1 の出力に対し、図 3.3 では右側 2 列が追加されているが、zeroRT 列と shftRT 列を較べると正確に 5%シフトされたレートとなっている。

また DF 列は小数点以下 3~4 術程度で違いが判断でき、tonaDF 列のような不便は無くなっている。では簡単にコードを概観しておく。

2.3: ZeroSpreadedTermStructure : イールドカーブをシフトさせるには ZeroSpreadedTermStructure コンストラクタを使用し、シフト後の新しいイールドカーブオブジェクト tnSFTOBJ を作成する。

ZeroSpreadedTermStructure Class Reference

Public Member Functions

```
ZeroSpreadedTermStructure (Handle<YieldTermStructure>, Handle<Quote> spread, Compounding comp=Continuous,
Frequency freq=NoFrequency, DayCounter dc=DayCounter())
```

- 上図はこのコンストラクタのリファレンスマニュアルのスクリーンショット。
- 引数は 5 個で、1 番目はシフト前のカーブハンドル tnCrvHDL、2 番目はシフト幅 5% のクオートハンドル。

- 残りの 3 つは TONA の市場慣例の cmpdSPL, freqA, dcA365 を与える。^{*5}

4, 5: tnSFtHDL, tnSFtIX : シフト用のカーブハンドルと TONA 指数を用意。

- 5 行目の OvernightIndex の最後の引数はシフトしたカーブハンドル tnSFtHDL を設定。

6, 7: dfSFT : シフト後のゼロレートと DF で構成されるデータフレーム。

- 7 行目 リスト内包表記のイテレータは図 3.1 で作成した dfTONA の date 列。
- この日付を使ってゼロレートと DF を算出。

8, 9: concat([dfTONA, dfSFT], axis=1) : dfTONA と dfSFT を横に結合し、出力。(concat に関しては “concat のまとめ, 136 ページ” を参照。)

11 : 図 3.1 で作成した TONA カーブと TONA 指数を 5% シフトしたオブジェクトに置き換えた。(図 3.3 のセルを無くしても 3.1.3 節以降のコードが動くように 3 つのオブジェクト名を同じにした)

3.1.3 TONA スワップ

| | |
|----|--|
| 1 | # 3. スワップ条件 payRcv: ql.OvernightIndexedSwap.Payer = 1 |
| 2 | effDT, matDT, payRcv, payLag, ntlAMT, cpnRT, sprdRT = ¥ |
| 3 | jDT(2022, 8, 23), jDT(2022, 8, 30), 1, 2, 10_000_000, 5.0/100, 0.0 |
| 4 | |
| 5 | # 4. スケジュール及びスワップオブジェクトの作成、エンジン設定 |
| 6 | fixSCD = ql.Schedule(effDT, matDT, pdFreqA, calJP, mFLLW, mFLLW, dtGENb, EoMf) |
| 7 | swapOBJ = ql.OvernightIndexedSwap(|
| 8 | payRcv, ntlAMT, fixSCD, cpnRT, dcA365, tonalIX, sprdRT, payLag) |
| 9 | swapOBJ.setPricingEngine(ql.DiscountingSwapEngine(tnCrvHDL)) |
| 10 | |
| 11 | # 5. 計算結果 |
| 12 | pd.DataFrame([|
| 13 | ['固定レグ時価', swapOBJ.legNPV(0)], ['変動レグ時価', swapOBJ.legNPV(1)], |
| 14 | ['スワップ時価 NPV', swapOBJ.NPV()], ['フェアレート', swapOBJ.fairRate()], |
| 15 | ['フェアスプレッド', swapOBJ.fairSpread()]], columns=['計算結果', ''], style.format(['':{:.7f}'])) |
| 16 | |

| 計算結果 | | 2 スワップ時価 NPV | -29.3591611 |
|----------|---------------|--------------|-------------|
| 0 固定レグ時価 | -9572.0373340 | 3 フェアレート | 0.0498466 |
| 1 変動レグ時価 | 9542.6781729 | 4 フェアスプレッド | 0.0001534 |

図 3.4: 1 週間 TONA スワップ

^{*5}QLP ドキュメントのコンストラクタの説明では “ql.ZeroSpreadedTermStructure(YieldTermStructure, spread)” で引数が 2 つの記載となる。しかし このコンストラクタの初期値は ql.Continuous(連続複利) であり、cmpdSPL を省略すると、スプレッドに与えた 5% は連続複利となる点に注意。

TONA カーブの構築を終え、図 3.4 によって、テナー 1 週間 (1w), クーポン 5.0% の TONA スワップを評価させた。このコードは図 2.8 (57 ページ) とほぼ同じものであり、相違する主な部分は次の通り。

2, 3 : スワップの各パラメータを設定。

payRcv=1 : payRcv=1 は **OvernightIndexedSwap.Payer** を意味。

- QuantLib のペイヤースワップは 1, レシーバーは -1。(一般的に call=1, put=-1)

payLag=2 : OIS の 2 日ラグ用に payLag 変数に設定。

cpnRT=5/100 : 固定レグのクーポンをシフト幅と同じ 5.0% を設定。

7, 8: OvernightIndexedSwap : OIS 用のコンストラクタで QLP ドキュメントでは下図を表示。

```
ql.OvernightIndexedSwap(swapType, nominals, schedule, fixedRate, fixedDC, overnightIndex)
Optional params:
spread=0.0
paymentLag=0
paymentAdjustment=ql.Following()
paymentCalendar=ql.Calendar()
telescopicValueDates=false
Types:
ql.OvernightIndexedSwap.Receiver
swapType = ql.OvernightIndexedSwap.Receiver
nominal = 100
schedule = ql.MakeSchedule(ql.Date(15,6,2020), ql.Date(15,6,2021), ql.Period('1Y'), calendar=ql.TARGET())
fixedRate = 0.01
fixedDC = ql.Actual360()
overnightIndex = ql.Eonia()
ois_swap = ql.OvernightIndexedSwap(swapType, nominal, schedule, fixedRate, fixedDC, overnightIndex)
```

- 引数は最低 6 個となり、2.2.4 節で説明した VanillaSwap コンストラクタと比較すると、floatSchedule が無い。
- 2 日ラグとして、paymentLag の値を 2 に設定する為、8 行目で 2 つの引数 sprdRT と payLag を記入。(名前付き引数には未対応)

(TONA スワップ キャッシュフロー表)

次の図は myUtil モジュール (A.2.3 節参照) の swapCashFlow 関数に tnCr-vOBJ オブジェクトと swapOBJ オブジェクトを与え、固定レグ (3 番目の引数:0) と変動レグ (同:1) のキャッシュフロー表を出力させた。

金利交換はスワップ満期時の 1 回であり、各レグは 1 行のみ (インデックス 0 番を除く) となる。変動レグに関して 次の各数値を確認しよう。

| | | | | | | | | |
|---|---------------|------------|------------|------------|------|-----------|------------|----------------------|
| 1 # 固定レグ:0 | | | | | | | | |
| 2 fmtSCF['amount']=':,.4f' | | | | | | | | # fmtSCFのamount桁数 修正 |
| 3 dfFixCF = mu.swapCashFlow(swapOBJ, tnCrvOBJ, 0) | | | | | | | | |
| 4 dfFixCF.style.format(fmtSCF) | | | | | | | | |
| | nominal | accruStart | accruEnd | payDate | days | rate | amount | DF |
| 0 | nan | nan | nan | 2022-08-23 | nan | nan% | nan | 0.99945344 |
| 1 | 10,000,000.00 | 2022-08-23 | 2022-08-30 | 2022-09-01 | 7 | 5.000000% | 9,589.0411 | 0.99822675 |
| | # 変動レグ:1 | | | | | | | |
| 2 dfFltCF = mu.swapCashFlow(swapOBJ, tnCrvOBJ, 1) | | | | | | | | |
| 3 dfFltCF.style.format(fmtSCF) | | | | | | | | |
| | fixingDate | accruStart | accruEnd | payDate | days | rate | spread | amount |
| 0 | nan | nan | nan | 2022-08-23 | nan | nan% | nan% | nan |
| 1 | 2022-08-29 | 2022-08-23 | 2022-08-30 | 2022-09-01 | 7 | 4.984664% | 0.0000% | 9,559.6298 |
| | | | | | | | | 0.99822675 |

図 3.5: 1週間 TONA スワップ キャッシュフロー表

payDate : この日付はスワップ満期日 (2022 年 8 月 30 日) から 2 営業日後の 2022 年 9 月 1 日。

rate : この TONA スワップはプレインバニラな為、変動レグの rate 4.984664% と図 3.4 “フェアレート” で表示されるスワップレートとは同じ値。

- 変動レグの 4.984664% はデイリーコンパウンドによって算出され、その計算法は 3.1.4 節で説明。
- 図 3.4 “フェアレート” で表示されるスワップレートはフォワードスワップレートの式 (2.17) で算出され、下のセルで再現した。

| | | | | | | | | |
|--|--|--|--|--|--|--|--|--|
| 1 # フェアレート | | | | | | | | |
| 2 effDF = tnCrvOBJ.discount(effDT) | | | | | | | | |
| 3 matDF = tnCrvOBJ.discount(matDT) | | | | | | | | |
| 4 annFCT = mu.calcAnnuity(fixSCD, tnCrvOBJ) | | | | | | | | |
| 5 swpRT = (effDF - matDF) / annFCT | | | | | | | | |
| 6 print(f'effDF:{effDF:.8f}', f'matDF:{matDF:.8f}', f'annuity:{annFCT:.8f}', f'swapRT:{swpRT:.6%}') | | | | | | | | |

effDF:0.99945344 matDF:0.99849891 annuity:0.01914929 swapRT:4.984664%

図 3.6: フェアレートの計算例

$$S_{2 \text{ 日} \times 7 \text{ 日}} = \frac{\overbrace{0.99945344}^{\text{8月23日 } DF} - \overbrace{0.99849891}^{\text{8月30日 } DF}}{\underbrace{0.01914929}_{\text{アニュイティ}}} = \underbrace{4.984664\%}_{\text{フェアレート}}$$

- 分母の mu.calcAnnuity(…) は図 2.21 で myUtil モジュールに組み入れたアニュイティ計算関数。
- この数値例からも金利計算の際、2 日ラグは無視されていることを理解しよう。

amount : 9,559.6298 円は accrueEnd 日 (2022 年 8 月 30 日) の金利受け取り額。

$$\underbrace{9,559.6298 \text{ 円}}_{amount} = \underbrace{4.984664\%}_{rate} \times \underbrace{\frac{7 \text{ 日}}{365 \text{ 日}}}_{days} \times 1,000 \text{ 万円}$$

DF : ディスカウントファクター 0.99822675 はスワップ満期日ではなく、支払日 (payDate: 2022 年 9 月 1 日) の値。

- この DF は 9 月 1 日に発生するキャッシュフローを現在価値に割り引く際に使用され、次式のように図 3.4 “変動レグ時価” の値を得る。

$$\underbrace{9542.6781729 \text{ 円}}_{\substack{\text{変動レグ時価} \\ (\text{図 3.4})}} = \underbrace{9,559.6298 \text{ 円}}_{amount} \times \underbrace{0.99822675}_{DF \ (9 \text{ 月 } 1 \text{ 日})} \quad (3.3)$$

2 日ラグの取り扱いに関して、図 3.2 の “paymentLag=0” の説明をこの数值例と比較しよう。

(図 3.5、図 3.6 のコードへの補足)

図 3.5 の 2 行目 : 固定レグのセルの 2 行目は myABBR モジュールで定義した辞書型変数 fmtSCF を修正する例。

- **fmtSCF['amount']=':,.4f'** : amount 項目を (少数 2 桁から) 少数 4 桁へ修正。(図 3.1 の fmtSCF.update(amount=':.4f') でも良い)

図 3.6 の 6, 7 行目 : この行の print 文では書式を **format メソッド** では無く、**f 文字列** (フォーマット文字列) で表している。

f 文字列の記入法は format メソッドと同様、文字列の中に置換フィールド (1.2.2 節参照) を書くが、次の 2 点が異なる。

- **f'…'** のように文字列の先頭に f を記入。
- 置換フィールド { 变数名: .8f } の中のコロンの前に変数名を記入。

format メソッドに較べ、f 文字列のほうがコンパクトに書くことが出来る為、この章以降は f 文字列を使用する。

3.1.4 TONA フォワードレートとデイリーコンパウンド

TONA レートは毎日リセットされる金利であり、TONA スワップの変動レグは TONA(テナー 1 日) のフォワードレートのデイリーコンパウンドで評価される。

この 3.1.4 節ではデイリーコンパウンドを数値例で紹介する。この説明は RFR 関連の商品を扱う場合、必ず理解すべき計算法である。

(TONA フォワードレート)

図 3.4(95 ページ) の TONA スワップで計算に使われた TONA フォワードレートの 5 日分(8 月 23 日～8 月 29 日)を表示させる為、図 3.7 上側セルでは次の 2 点を修正した。

2: pdFreqD : 図 3.4 の Schedule コンストラクタの 3 番目引数を pdFreqA から、1 日単位を意味する **pdFreqD** へ修正。

| | # Schedule 3番目引数をpdFreqD, payLagは0へ |
|---|---|
| 1 | fixSCDdy = ql.Schedule(effDT, matDT, pdFreqD, calJP, mFLLW, mFLLW, dtGENb, EoMf) |
| 2 | swapOBJdy = ql.OvernightIndexedSwap(payRcv, ntIAMT, fixSCDdy, cpnRT, dcA365, tonalIX) |
| 3 | dfCFdy = mu.swapCashFlow(swapOBJdy, tnCrvOBJ, 1) # 変動レグ:1 |
| 4 | dfCFdy.style.format(fmtSCF) |
| 5 | |
| | fixingDate accrueStart accrueEnd payDate days rate spread amount DF |
| 0 | nan nan nan 2022-08-23 nan nan% nan% nan 0.99945344 |
| 1 | 2022-08-23 2022-08-23 2022-08-24 2022-08-24 1 4.984664% 0.000% 1,365.6614 0.99931697 |
| 2 | 2022-08-24 2022-08-24 2022-08-25 2022-08-25 1 4.983983% 0.000% 1,365.4749 0.99918053 |
| 3 | 2022-08-25 2022-08-25 2022-08-26 2022-08-26 1 4.983303% 0.000% 1,365.2885 0.99904413 |
| 4 | 2022-08-26 2022-08-26 2022-08-29 2022-08-29 3 4.982623% 0.000% 4,095.3064 0.99863516 |
| 5 | 2022-08-29 2022-08-29 2022-08-30 2022-08-30 1 4.980583% 0.000% 1,364.5433 0.99849891 |
| | # 8月29日のrate |
| 1 | aug29Fwd = (dfCFdy.DF[4]/dfCFdy.DF[5] -1)*365 |
| 2 | print(f'TONAレート(Aug29): {aug29Fwd:. 6%}') |
| 3 | |

TONA レート(Aug29): 4.980583%

図 3.7: TONA フォワードレートの計算例

3: payLag=0 : 図 3.4 の OvernightIndexedSwap コンストラクタの引数を 8 個から 6 個に減らし、paymentLag=0 の初期値に戻した。

- ラグが無いので accrueEnd と payDate は同じ日付。
- 同じ理由で DF 列は accrueEnd 日のディスカウントファクターを表示。

新たに作り直したスワップオブジェクト swapOBJdy (dy は daily の略) の変動レグキャッシュフローが図 3.7 上側セルの出力であり、TONA フォワードレートは式 (2.6) (75 ページ) で算出されている。

この点は同図下側セルで 8 月 29 日のフォワードレート 4.980583%を算出し、インデックス 5 番 rate 列と同じ値を得ている。

(デイリーコンパウンド)

デイリーコンパウンド (Daily compounded, 日次複利) の定義は“日々のレートで想定元本が毎日複利運用されること”であり、計算式は以下となる。(想定元本 1 円)

$$\begin{aligned} \text{将来価値} &= \underbrace{(1 + \tau_1 \times f_1) \times \cdots \times (1 + \tau_i \times f_i)}_{\text{コンパウンド部分}} - 1 & (3.4) \\ i &= 1, 2, \dots, n \text{ (リセット日)} \\ \tau_i &: i \text{ 日のテナー (休日を跨がない場合 1 日/365)} \\ f_i &: i \text{ 日満期のフォワードレート} \end{aligned}$$

この式の f_i に図 3.7 rate 列の各フォワードレートを、 τ_i に 365 日で割った days 列の日数を当てはめれば 变動レグの将来価値となる。下図でその計算を行った。

```
1 # 变动レグの将来価値とrate
2 dlyCmp = (1+ dfCFdy.days[1:]/365 * dfCFdy.rate[1:]).cumprod() -1
3 print(f'将来価値: {dlyCmp.iloc[-1]*ntlAMT:.4f}', 
      f'フェアレート: {dlyCmp.iloc[-1]*365/7 :.6%}' )
```

将来価値: 9,559.6298 フェアレート: 4.984664%

結果は図 3.5 (97 ページ) の変動レグの amount 列 9,559.6298 と同じ値を得ている。同時に 4 行目で同図 rate 列 4.984664%(フェアレート) を計算させた。では簡単に計算を見て行こう。

2: $(1 + \text{dfCFd.days}[1:] / 365 * \text{dfCFd.rate}[1:])$:

- 式 (3.4) の $(1 + \tau_i \times f_i)$ を計算。
- $\text{dfCFd.days}[1:] / 365$ 部分が τ_i , $\text{dfCFd.rate}[1:]$ が f_i 。

2: $(\dots).cumprod()$: (\dots) 部分は Pandas の **Series** となっている。cumprod はそのメソッドで累積積を計算。

- 下セルは [3, 4, 5] という Series に cumprod で累積積を計算した例。

```
1 # cumprodとiloc[-1]
2 srEX1 = pd.Series([3, 4, 5]).cumprod() ; print(' cumprod= ¥n', srEX1)
3 print(' iloc[-1]=', srEX1.iloc[-1])
```

| | | |
|------------|----|--------------|
| cumprod= 0 | 3 | iloc[-1]= 60 |
| 1 | 12 | |
| 2 | 60 | |

- 累積積の計算結果は [3, 12, 60] という Series。
- 最後の要素で計算される $3 \times 4 \times 5 = 60$ を取り出すには **iloc[-1]** が必要。(Pandas は [-1] というスライスが出来ず、iloc メソッドが必要)

3: iloc[-1]* ntlAMT : デイリーコンパウンドを計算した dlyCmp 変数は Pandas Series なので iloc[-1] により、最後の要素を抽出し、ntlAMT を掛けて 将来価値を算出。

4: フェアレート : 式 (3.4) は利息を計算しているので、スワップのテナー (7/365) で割って年率化することで図 3.4 (95 ページ) のフェアレート 4.98466% を算出。

尚、図 3.7 で作成したスワップオブジェクト swapOBJdy は次の 2 点で本来の TONA スワップの評価が出来ない点に注意。

- 2 日ラグが未設定。
- 金利交換を Daily(pdFreqD) にした為、デイリーコンパウンドの計算が行われない。

ただし pdFreqA から pdFreqD への修正等で、TONA フォワードレートの値やその計算根拠となるディスカウントファクター等が判るのは非常に便利である。

3.1.5 TONA スワップの期中評価

この 3.1.5 節では日付が 8 月 25 日まで進んだとして、図 3.4 の 1 週間 TONA スワップを評価し、変化した数値を確認する。(ただし、イールドカーブは不变)

図 3.8 が評価用コードと結果となり、以下の点を確認しよう。

2~5: addFixings : 2.4.4 節で説明したように変動レグの指標金利にフィクシングデータをセットしないと図 3.4 の swapOBJ はエラーを戻す。

2,3 : フィクシングデータとして 8 月 23 日 5.1%、8 月 24 日 5.2% を準備。

4 : addFixings メソッドで tonaIX へセット。(図 2.15 参照)

5 : QuantLib の日付を 8 月 25 日に設定。

以上の準備によって、これまでのコードは修正の必要無く 期中評価を行うことが出来る。

```

1 # 8月25日評価
2 tnFixDT = [jDT(2022,8,23), jDT(2022,8,24)]           # Tona fixing
3 tnFixRT = [ 5.1/100, 5.2/100 ]
4 tonalX.addFixings(tnFixDT, tnFixRT, True)
5 tradeDT = jDT(2022,8,25) ; setEvDT(tradeDT)          # 日付修正
6
7 print(' (8月25日評価)')                                # スワップ評価
8 display( pd.DataFrame([
9     [' 固定レグ時価', swapOBJ.legNPV(0)],   [' 変動レグ時価', swapOBJ.legNPV(1)],
10    [' スワップ時価 NPV', swapOBJ.NPV()],      [' フェアレート', swapOBJ.fairRate()],
11    [' フェアスプレッド', swapOBJ.fairSpread()]], columns=[' 計算結果', '']) )
12 .style.format({'':{:.7f}}) )
13 print(' (変動レグ明細)')                                # 変動レグ
14 display( mu.swapCashFlow(swapOBJ, tnCrvOBJ, 1).style.format(fmtSCF) )
15
16 print(' (TONAフォワードレート明細)')                  # TONAフォワードレート
17 dfAug25 = mu.swapCashFlow(swapOBJdy, tnCrvOBJ, 1)
18 display( dfAug25.style.format(fmtSCF) )                   # 変動レグの将来価値とrate
19
20 dlyCmp = (1+ dfAug25.days[1:]/365 * dfAug25.rate[1:]).cumprod()-1
21 print(f'(hc)将来価値: {dlyCmp.iloc[-1]*ntlAMT:.4f}', 
22 f'(hc)フェアレート: {dlyCmp.iloc[-1]*365/7 :.6%} ')

```

| (8月25日評価) | | 計算結果 | | 2 スワップ時価 NPV | 67.9692499 |
|-----------|--------|---------------|---|--------------|------------|
| 0 | 固定レグ時価 | -9579.8771324 | 3 | フェアレート | 0.0503548 |
| 1 | 変動レグ時価 | 9647.8463823 | 4 | フェアスプレッド | -0.0003548 |

(変動レグ明細)

| fixingDate | accruStart | accruEnd | payDate | days | rate | spread | amount | DF |
|------------|------------|------------|------------|------------|------|-----------|--------|-----------------------|
| 0 | nan | nan | nan | 2022-08-23 | nan | nan% | nan% | nan 1.00000000 |
| 1 | 2022-08-29 | 2022-08-23 | 2022-08-30 | 2022-09-01 | 7 | 5.035475% | 0.000% | 9,657.0754 0.99904433 |

(TONAフォワードレート明細)

| fixingDate | accruStart | accruEnd | payDate | days | rate | spread | amount | DF |
|------------|------------|------------|------------|------------|------|-----------|--------|-----------------------|
| 0 | nan | nan | nan | 2022-08-23 | nan | nan% | nan% | nan 1.00000000 |
| 1 | 2022-08-23 | 2022-08-23 | 2022-08-24 | 2022-08-24 | 1 | 5.100000% | 0.000% | 1,397.2603 1.00000000 |
| 2 | 2022-08-24 | 2022-08-24 | 2022-08-25 | 2022-08-25 | 1 | 5.200000% | 0.000% | 1,424.6575 1.00000000 |
| 3 | 2022-08-25 | 2022-08-25 | 2022-08-26 | 2022-08-26 | 1 | 4.991000% | 0.000% | 1,367.3973 0.99986328 |
| 4 | 2022-08-26 | 2022-08-26 | 2022-08-29 | 2022-08-29 | 3 | 4.986708% | 0.000% | 4,098.6642 0.99945364 |
| 5 | 2022-08-29 | 2022-08-29 | 2022-08-30 | 2022-08-30 | 1 | 4.984665% | 0.000% | 1,365.6617 0.99931716 |

(hc)将来価値: 9,657.0754 (hc)フェアレート: 5.035475%

図 3.8: 8月25日での1週間TONAスワップ評価

7~12: (8月25日評価)と図3.4の比較:

固定レグ時価: 8月19日とほぼ同じ値を算出。

- 若干の変化はキャッシュフロー日(9月1日)のディスカウントファクターが変わった為。

変動レグ時価: この金額を確認するにはデイリーコンパウンドを手計算しなければならない。(将来価値とフェアレートは16行以下で算出)

- 金額的には100円程大きい。

- その理由はカーブにある TONA レート (図 3.3 の 4.991%) よりもフィクシングレート 5.1%、5.2% を大きくした為。

フェアレート：上の“変動レグ時価”と同じ理由で若干高い。

13, 14: (変動レグ明細) の出力：amount 列 9,657.0754 は将来価値で、DF 列を掛ければ、変動レグ時価 9467.8464 を得るが、amount 列の金額を 16 行目以下で算出。

16～22: 変動レグの将来価値と金利の手計算：

17,18：デイリーコンパウンドの手計算の為、図 3.7 と同様に swapOB-Jdy オブジェクトで TONA フォワードレートのデータフレームを dfAug25 にセット。

20：式 (3.4) でデイリーコンパウンドの手計算。(100 ページのコードとほぼ同じ)

21,22：変動レグの将来価値 9,657.0754 とフェアレート 5.035475% を算出。

- 2 つの値は 14 行目で出力した“(変動レグ明細)”の値と一致。

3.2 SOFR カーブと SOFR スワップ

2023 年 7 月から US ドル Libor は公表停止となり、US ドル金利のデリバティは SOFR を参照する商品へ移行した。この節ではその中心を成す SOFR スワップカーブ (以下 **SOFR カーブ**) と **SOFR スワップ**を算出する QuantLib のコードを紹介する。

3.2.1 SOFR カーブ

図 3.9 は SOFR カーブ作成のコードと出力されたディスカウントファクターである。このコードは TONA カーブの図 3.1 とほぼ同じなので、相違点のみを以下で確認する。(“セル”に続く数字はコードの行番号を示す)

1 番セル: カーブデータ crvDATA :

カーブデータ crvDATA は CME の Web [19], [20] 等を参考に著者が作成した架空の 2023 年 9 月 26 日の SOFR レート。(このカーブは逆イールドだがマイナス金利では無く、ある程度の実勢を表している)

3 番セル 4: データフレーム dfSOFR :

4 行目データフレーム dfSOFR はゼロレート等の計算はせず、ディスカウントファクターのみを出力。

```

1 from myABBR import *; import myUtil as mu
2 crvDATA = [ ('depo', '1d', 5.31), ('swap', '1m', 5.32), ('swap', '3m', 5.38),
3             ('swap', '6m', 5.46), ('swap', '1y', 5.45), ('swap', '2y', 5.01),
4             ('swap', '3y', 4.67) ]
5
6 def makeSofrCurve(crvDATA):
7
8     # 1. 指標金利オブジェクトと初期値設定
9     sfCrvHDL = ql.RelinkableYieldTermStructureHandle()
10    sofrIX = ql.Sofr(sfCrvHDL)
11
12     # 2. HelperとSOFRカーブオブジェクト
13     cHelper, sfParRATE = [], []
14
15     for knd, tnr, rt in crvDATA:
16         if knd == 'depo':
17             cHelper.append(ql.DepositRateHelper(mu.sqHDL(rt/100), sofrIX))
18         if knd == 'swap':
19             cHelper.append(ql.OISRateHelper(Tp2, pD(tnr), mu.sqHDL(rt/100), sofrIX))
20             sfParRATE.append(rt/100) # パーレート用リスト
21
22     # カーブオブジェクト
23     sfCrvOBJ = ql.PiecewiseLogLinearDiscount(Tp0, calIUSs, cHelper, dcA360)
24     sfCrvHDL.linkTo(sfCrvOBJ); sfCrvOBJ.enableExtrapolation()
25
26     return [sofrIX, sfCrvOBJ, sfCrvHDL, sfParRATE] # 4つのオブジェクトを戻す
27
28
29 tradedT = jDT(2023, 9, 26); setEvDT(tradeDT)
30 # SOFRカーブ作成
31 sofrIX, sfCrvOBJ, sfCrvHDL, sfParRATE = makeSofrCurve(crvDATA)
32 dfSOFR = pd.DataFrame([ (dt.ISO(), df) for dt, df in sfCrvOBJ.nodes() ],
33                         columns=[ 'date', 'DF' ])
34
35 print("決済日(reference): ", sfCrvOBJ.referenceDate().ISO())
36 dfSOFR.style.format({'DF': '{:.9f}' })

```

決済日(reference) : 2023-09-26

| | date | DF | 2 | 2023-10-30 | 0.994999881 | 5 | 2024-09-30 | 0.946949515 |
|----------|-------------|-------------|----------|------------|-------------|----------|------------|-------------|
| 0 | 2023-09-26 | 1.000000000 | 3 | 2023-12-28 | 0.986292100 | 6 | 2025-09-29 | 0.905346831 |
| 1 | 2023-09-27 | 0.999852522 | 4 | 2024-03-28 | 0.972851189 | 7 | 2026-09-28 | 0.870639746 |

図 3.9: SOFR カーブ

2番セル: makeSoftCurve 関数 : (この関数は myUtil モジュールに追加)

5: Sofr コンストラクタ : SOFR 指数 (及び ESTR 指数) は Sofr コンストラクタ (**Estr** コンストラクタ) によって、SOFR 指数オブジェクト **sofrIX** を作成。

Sofr Class Reference

```
#include <ql/indexes/ibor/sofr.hpp>
```

Inherits OvernightIndex

Public Member Functions

Sofr (const Handle< YieldTermStructure > &h=0)

- 上図は Sofr クラスのリファレンスマニュアル (1.6 節参照) で、最後の行から、Sofr コンストラクタの引数が YieldTermStructure(YTS) ハンドルの 1 つのみ。(図 3.1 の OvernightIndex コンストラクタでは引数 6 個)
 - この図の 2 行目の “ql/indexes/ibor/sofr.hpp” は SOFR クラスを記述した C++ のコードが QuantLib のソースコードリス

ト [46] のどこにあるかを示している。⁶

- 3 行目に “Inherits OvernightIndex” とあり、Sofr クラスは OvernightIndex クラス (図 3.1 の TONA カーブで使用) を継承。(継承を遡ると、第 1 章の Tibor クラスのメソッドや 3.1 節で例示した fixing 関連のメソッド等は使用可能が判る)

15: calUSs, dcA360 : TONA カーブと異なり、図 3.9 の 15 行目カーブコンストラクタには calUSs 及び dcA360 を設定。

- calUSs は SOFR フィクシングカレンダーの短縮形で、calUSs = ql.UnitedStates(ql.UnitedStates.SOFR) を myABBR モジュールで定義。(A.1 節参照)
- 短縮形 dcA360 = ql.Actual360() を myABBR モジュールで定義。

(補足: SOFR フィクシングカレンダー)

Sofr フィクシング関連のデータを確認する場合、sofrIX オブジェクトを使い、図 1.16(30 ページ) と同じメソッドを用いれば良い。(メソッドは継承されている)

```

1 settleDT = sofrIX.valueDate(tradeDT)
2 print('trade      : ', tradeDT.ISO())
3 print('settle     : ', settleDT.ISO())
4 print('-----')
5 print('fixingDays : ', sofrIX.fixingDays())
6 print('fixingDate  : ', sofrIX.fixingDate(settleDT).ISO(), '\n')
7 print('tenor       : ', sofrIX.tenor())
8 print('dayCounter  : ', sofrIX.dayCounter())
9 print('fixingCalendar : ', sofrIX.fixingCalendar())
10 print('maturityDate : ', sofrIX.maturityDate(settleDT).ISO())

```

| | | | |
|------------|--------------|----------------|---------------------------------|
| trade | : 2023-09-26 | tenor | : 1D |
| settle | : 2023-09-26 | dayCounter | : Actual/360 day counter |
| ----- | ----- | fixingCalendar | : SOFR fixing calendar calendar |
| fixingDays | : 0 | maturityDate | : 2023-09-27 |
| fixingDate | : 2023-09-26 | | |

図 3.10: Sofr オブジェクトの属性

- 9 行目 fixingCalendar メソッドの出力は “SOFR fixing calendar”。
- これは短縮形 calUSs として定義したカレンダークラス **UnitedStates** に引数 ql.UnitedStates.SOFR を与えて作成するカレンダーオブジェクトを意味。

3.2.2 SOFR スワップ

図 3.11 は 2 年 SOFR スワップのコード例であり、TONA スワップ (図 3.4) からの修正点は 6 行目と 8 行目のカレンダーと日数計算法となる。

⁶Sofr クラスは新しく作成された為、ドキュメントが間に合っていない。そのような場合、C++ のコードで確認する必要がある。

```

1 # スワップ条件 payRcv:ql.OvernightIndexedSwap.Payer = 1
2 effDT, matDT, payRcv, payLag, ntlAMT, cpnRT, sprdRT =¥
3 jDT(2023, 9, 28), jDT(2025, 9, 28), 1, 2, 10_000_000, 5.0/100, 0.0
4
5 # スケジュール及びスワップオブジェクトの作成、エンジン設定
6 fixSCD = ql.Schedule(effDT, matDT, pdFreqA, calUSf, mFLLW, mFLLW, dtGENb, EoMf)
7 swapOBJ = ql.OvernightIndexedSwap(
8     payRcv, ntlAMT, fixSCD, cpnRT, dcA360, sofrIX, sprdRT, payLag)
9 swapOBJ.setPricingEngine(ql.DiscountingSwapEngine(sfCrvHDL))
10
11 # 計算結果
12 pd.DataFrame([
13     ['固定レグ時価', swapOBJ.legNPV(0)], ['変動レグ時価', swapOBJ.legNPV(1)],
14     ['スワップ時価 NPV', swapOBJ.NPV()], ['フェアレート', swapOBJ.fairRate()],
15     ['フェアスプレッド', swapOBJ.fairSpread()]], columns=['計算結果', ''])
16 .style.format({'': '{:.5f}'})

```

| | 計算結果 | 2 スワップ時価 NPV | 1881.59771 |
|---|----------------------|--------------|------------|
| 0 | 固定レグ時価 -941481.78212 | 3 フェアレート | 0.05010 |
| 1 | 変動レグ時価 943363.37982 | 4 フェアスプレッド | -0.00010 |

図 3.11: 2 年 SOFR スワップ

6: calUSf : この行では固定レグのスケジュールを作成するので、カレンダー

は Fedwire 決済 (フェドワイヤー) の calUSf を使用。(短縮形 calUSf=ql.UnitedStates(ql.UnitedStates.FederalReserve) を myABBR モジュールで定義)

- 前節の SOFR フィクシングカレンダー calUSS と Fedwire 決済カレンダーは異なる点に注意。

```

1 # 休日リスト (SOFRとFederalReserveの比較)
2 print( calUSS.holidayList(jDT(2023, 1, 1), jDT(2023, 12, 31))[:6] )
3 print( calUSf.holidayList(jDT(2023, 1, 1), jDT(2023, 12, 31))[:6] )

(Date(2, 1, 2023), Date(16, 1, 2023), Date(20, 2, 2023), Date(7, 4, 2023), Date(29, 5, 2023), Date(19, 6, 2023))
(Date(2, 1, 2023), Date(16, 1, 2023), Date(20, 2, 2023), Date(29, 5, 2023), Date(19, 6, 2023), Date(4, 7, 2023))

```

- 上図はそれぞれのカレンダーの **holidayList** メソッドで休日リストを表示。
- 4 月 7 日 (Good Friday) では SOFR はフィクシングされないが、Fedwire は動いている為 決済可能となり、calUSf では休日とならない。

8: dcA360 : SOFR スワップ固定レグの日数計算法は変動レグと同じ dcA360。

図 3.12 の上側セルは変動レグのキャッシュフロー表であり、コード的には図 3.5 と同じであるが、2 年のスワップなので出力は 3 行となっている。

| | # 変動レグ:1 |
|---|--|
| 0 | dfFltCF = mu.swapCashFlow(swapOBJ, sfCrvOBJ, 1, dcA360) |
| 1 | dfFltCF.style.format(fmtSCF) |
| 2 | |
| | fixingDate accrueStart accrueEnd payDate days rate spread amount DF |
| 0 | nan nan nan 2023-09-28 nan nan% nan% nan 0.99970512 |
| 1 | 2024-09-27 2023-09-28 2024-09-30 2024-10-02 368 5.450000% 0.000% 557,111.11 0.94671578 |
| 2 | 2025-09-26 2024-09-30 2025-09-29 2025-10-01 364 4.544724% 0.000% 459,522.06 0.90515240 |
| | |
| 1 | # forward swap rate |
| 2 | effDF = sfCrvOBJ.discount(iDT(dfFltCF.accruEnd[1])) |
| 3 | matDF = sfCrvOBJ.discount(iDT(dfFltCF.accruEnd[2])) |
| 4 | annFCT = 364/360*matDF |
| 5 | print(f'forward swap rate: {(effDF-matDF)/annFCT:.6%}') |
| | forward swap rate: 4.544724% |

図 3.12: 2 年 SOFR スワップの変動レグ キャッシュフロー表

このキャッシュフロー表から確認すべき点が rate 列に表示されている次の 2 つの金利。

5.450000% : 図 3.9 の 1 番セルで与えられた 1 年 SOFR スワップレート。

4.544724% : 同図 1 年と 2 年の SOFR スワップレートから計算された 1 年先 1 年のフォワードスワップレート。

- 図 3.12 下側セルは式 (2.17)(79 ページ) で手計算し、同じレートを算出。(数値式は省くので各自確認)
- 算出に当たり 注意すべき点は変動レグのキャッシュフロー表の DF 列で表示されるディスカウントファクターは 2 日ラグの DF。
- ラグの無い DF の値は下側セル 2 行目のように sfCrvOBJ.discount(iDT(dfFltCF.accruEnd[1])) で取得。(この SOFR スワップの満期日 2025 年 9 月 28 日は日曜)

3.3 RFR タームレートとベーススカーブ*

TONA や SOFR 等の RFR はオーバーナイトレート(翌日物金利)であり、後決め金利であるが、図 3.1 や図 3.9 の 1 番セルのスワップレートはオーバーナイトレートの対義語としてタームレートと呼ばれ、前決め金利である。

実際 現在の 1 年スワップレートは 5.450000% として前決めされ、取引が行われている。1 年先 1 年のフォワードレート 4.544724% も前決め金利である。

ただこれらの金利^{*7}の確定はキャッシュフロー表のfixingDate列の日付まで待つ必要がある。このことは2年先の6ヶ月Tiborレートは現在のTiborカーブから算出することが出来るが、このTiborのfixingは2年後(2.5年後ではなくいが)となることに似ている。

前節の最後でRFRを指標金利とするスワップレートは前決め金利である旨を伝えた。後決め金利は扱い難い為、3ヶ月物RFRスワップレートやRFR先物(第7章参照)は旧3ヶ月USDLiborに代わる指標金利の役割を担い始めている。同時に3ヶ月SOFR先物やスワップレートを変動レグの指標金利とするスワップ取引が拡大しつつある。

3.3.1 CME 3ヶ月 TermSOFR カーブ

CMEはSOFR先物の取引データから3ヶ月タームSOFR(3monthTermSOFR)レートを公表している。^{*8}

ここでは3ヶ月タームSOFRを変動レグの指標とするスワップカーブ(以下3mTermSOFRカーブ)を構築する。(TermSOFRと区別する場合、オーバーナイトレートのSOFRはOvernightSOFRと言う)

CME TermSOFRを指標金利とするスワップは3.2.1節で説明したOvernight-SOFRカーブとのベーシススプレッド(basis spread)で建値される。^{*9}

図3.13は3ヶ月TermSOFRカーブ構築のコード例であり、1行～7行目で図3.9と同じSOFRカーブを作成している。(myUtilモジュールに追加したmakeSofrCurve関数を利用)

TermSOFRカーブの構築は9行目から始まり、11行目でベーシスデータを設定している。ただこのベーシス情報はブローカー間のデータとなり、一般的なWebでは公表されていない為、全てのノードをゼロとした。^{*10}

OvernightSOFRカーブの図3.9と計算結果を比較すると、ゼロベーシスの為各ノードのディスカウントファクターは小数点以下4桁目辺りで若干異なる程度となっている。

^{*7}この章の脚注^{*1}に記したLyashenko and Mercurio [41]が前決めと後決めを統一的に扱うフレームワークを示した資料である。

^{*8}算出方法はCME [21]を参照。

^{*9}JSCCではTIBORカーブがTONAカーブとのベーシス(このベーシスをテナーベーシスと言う)で建値されていることに類似。

^{*10}情報端末BloombergのSOFR vs 3m CME Term SOFR Basis 3年のティッカーはUSSVTQ3。

```

1 from myABBR import *      ; import myUtil as mu
2 tradeDT = jDT(2023, 9, 26) : setEvDT(tradeDT)
3 # 0. SOFRカーブ
4 crvDATA = [('depo', '1d', 5.31), ('swap', '1m', 5.32), ('swap', '3m', 5.38),
5             ('swap', '6m', 5.46), ('swap', '1y', 5.45), ('swap', '2y', 5.01),
6             ('swap', '3y', 4.67)]
7 sofrIX, sfCrvOBJ, sfCrvHDL, sfParRATE = mu.makeSofrCurve(crvDATA)
8
9 # 1. CME Term SOFR rate and Basis curve (= all zero)
10 TsfRT3m = 5.38558
11 TsfCrvBS = [('6m', 0.0), ('1y', 0.0), ('2y', 0.0), ('3y', 0.0)]
12
13 # 2. TermSOFR指標金利オブジェクト
14 TsfCrvHDL = ql.RelinkeableYieldTermStructureHandle()
15 TsfIX = ql.IborIndex('TermSofr', pdFreqQ, Tp2, usdFX, calUSs, mFLLW,
16                         EoMt, dcA360, TsfCrvHDL)
17 # 3. Basis helperでのTermSOFRカーブオブジェクト
18 cHelper = [ql.DepositRateHelper(mu.sqHDL(TsfRT3m/100), TsfIX)]
19 for tnr, bs in TsfCrvBS:
20     cHelper.append(ql.OVERNIGHTIBORBASISSWAPRATEHELPER(mu.sqHDL(bs/100),
21                                                       pD(tnr), Tp2, calUSs, mFLLW, EoMt, sofrIX, TsfIX, sfCrvHDL))
22 TsfCrvOBJ = ql.PiecewiseLogLinearDiscount(Tp2, calUSs, cHelper, dcA360)
23 TsfCrvHDL.linkTo(TsfCrvOBJ) : TsfCrvOBJ.enableExtrapolation()
24 # checking
25 print('決済日(reference):', TsfCrvOBJ.referenceDate().ISO())
26 [(dt.ISO(), df) for dt, df in TsfCrvOBJ.nodes()]

```

決済日(reference): 2023-09-28
[('2023-09-28', 1.0), ('2024-09-30', 0.947229008881084),
('2023-12-28', 0.986569201876598), ('2025-09-29', 0.9056140456298556),
('2024-03-28', 0.9731383278012513), ('2026-09-28', 0.8708967171925004)]

図 3.13: TermSOFR カーブの構築例

このコードは通常のカーブ構築コードとほぼ同じであり、以下 注意点のみを記す。

10: TsfRT3m = 5.38558 : CME が発表した 2023 年 9 月 26 日の 3ヶ月 TermSOFR レートは 5.38558%。 (TsfRT は TermSofr rate の略)

- 4 行目で与えた 3ヶ月 SOFR スワップレートは 5.38% であり、若干異なる。

11: TsfCrvBS : OvernightSOFR カーブとのベースを%単位で設定。(ここでベースはすべてゼロとした)

- このリストに 3ヶ月のベース ('3m', 0.0) を入れると “more than one instrument with pillar December 28th, 2023” というエラーになる点に注意。(2023 年 12 月 28 日満期の証券が 2 つになる為)

15, 16: IborIndex : 3ヶ月 TermSOFR 金利指数を IborIndex コンストラクタより作成。

IborIndex

```
ql.IborIndex(familyName, tenor, settlementDays, currency, fixingCalendar, convention, endOfMonth,
dayCounter, =Handleql.YieldTermStructure())

ql.IborIndex('MyIndex', ql.Period('6m'), 2, ql.EURCurrency(), ql.TARGET(), ql.ModifiedFollowing, True,
ql.Libor('MyIndex', ql.Period('6M'), 2, ql.USDCurrency(), ql.TARGET(), ql.Actual360()      ql.Actual360())
ql.Euribor(ql.Period('6M'))
ql.USDLibor(ql.Period('6M'))
ql.Euribor6M()
```

図 3.14: IborIndex コンストラクタ

引数 : QLP ドキュメントでは、引数は次の 9 個。大半は説明済みの引数。

familyName, tenor, settlementDays, currency, fixingCalendar, convention, endOfMonth, dayCounter, Handle.YieldTermStructure

15: usdFX : 通貨を表し、myABBR モジュールでは usdFX=ql.**USDCurrency()** と定義。

16: EoMt : USDLibor に倣い、ここではendOfMonth は True に設定。^{*11}

20, 21: OvernightIborBasisSwapRateHelper : RFR カーブとのベースから イールドカーブを作成するヘルパーを準備。

OvernightIborBasisSwapRateHelper Class Reference**Public Member Functions**

```
OvernightIborBasisSwapRateHelper(const Handle< Quote > &basis, const Period &tenor, Natural settlementDays, Calendar calendar,
BusinessDayConvention convention, bool endOfMonth, const ext::shared_ptr< OvernightIndex > &baseIndex, const ext::shared_ptr<
IborIndex > &otherIndex, Handle< YieldTermStructure > discountHandle=Handle< YieldTermStructure >())
```

引数 : リファレンスマニュアルでは、引数は次の 9 個。

basis, tenor, settlementDays, calendar, convention, endOfMonth, baseIndex, otherIndex, Handle.YieldTermStructure

baseIndex : 7 行目の sofrIX を指定。

otherIndex : 15 行目の TsfIX を指定。

尚 この 3ヶ月 TermSOFR カーブは 7.3 節で説明する 3ヶ月 TermSOFR 金利キャップを評価する際に使用する。

3.4 マルチカーブとブートストラップ法*

初めにシングルカーブのブートストラップについて説明し、次にマルチカーブに移行する。

*¹¹CME の資料では EoM の扱いは確認出来なかったが、旧 Libor は True であった。

金利分野でのブートストラップ法は利付債やスワップレート等、満期前に利払い等のキャッシュフローがある金利商品から、満期時点のディスカウントファクター（以下 D_E ）を計算する方法を言う。（この名称の由来は脚注¹² 参照）

次式は式 (2.13) において D_E を左辺にまとめている。

$$(1 + \tau_E S)D_E = 1 - S \sum_{i=1}^{E-1} (\tau_i \times D_i) \quad (3.5)$$

ブートストラップ法の目的は D_E について解くことであり、式 (3.5) より

$$D_E = \frac{1 - S \sum_{i=1}^{n-1} (\tau_i \times D_i)}{(1 + \tau_E S)} \quad (3.6)$$

を得る。

3.4.1 シングルカーブのブートストラップ

最も単純な例から始めよう。トイモデル (toy model) として 1 年指標金利 7.0% (1 年 Tibor を想定)、2 年スワップ 13.0% (Annual) の 2 つのノードのみで構成するイールドカーブとする。

図 3.15 はこのカーブのオブジェクト sCrvOBJ を作成し、2 年ディスカウントファクター 0.7774377636 を QuantLib と手計算で算出したコードである。

以下 コードと計算を説明する。

1: setEvDT(jDT(2021,8,2)) : EvaluationDate (取引日) を 2021 年 8 月 2 日にすることで、1 年、2 年の日付は 2022 年 8 月 4 日、2023 年 8 月 4 日となり、Actual365 日数計算法で各テナーは 1.0。（計算式で 1.0 は省略）

5, 6: Rt1y, Rt2y, IX1y : 1 年 Tibor 7.0% 用に Rt1y、2 年スワップ 13.0% 用に Rt2y を設定。（Rt は rate を意味）

- IX1y は変動レグの指標金利用変数。（IX は index の略）
- ここでは Tibor コンストラクタを使ったが、指標金利用コンストラクタなら、いずれでも可。

8, 9: sHelper : 2 つの金利でヘルパー sHelper を作成。（s は single を意味）

- Rt2y は Annual のスワップレートの為、freqA を引数に指定。
- Rt2y は 2022 年 8 月 4 日、2023 年 8 月 4 日でキャッシュフローがある為 ブートストラップが必要。

```

1 from myABBR import * ; import myUtil as mu ; setEvDT(jDT(2021,8,2))
2
3 # Forward curve 金利
4 sCrvHDL = ql.RelinkableYieldTermStructureHandle()
5 Rt1y, Rt2y, IX1y = ql.Tibor(pdFreqA, sCrvHDL)
6 0.07, 0.13, IX1y = ql.Tibor(pdFreqA, sCrvHDL)
7 # カーブ構築
8 sHelper = [ql.DepositRateHelper(Rt1y, IX1y),
9             ql.SwapRateHelper(Rt2y, pd('2y'), calJP, freqA, mFLLW, dcA365, IX1y)]
10 sCrvOBJ = ql.PiecewiseLogLinearDiscount(Tp2, calJP, sHelper, dcA365)
11 sCrvHDL.linkTo(sCrvOBJ)
12 # DF表示
13 dfSG = pd.DataFrame([(dt.ISO(), df, sCrvOBJ.zeroRate(dt, dcA365, cmpdSPL, freqA).rate())
14                       for dt, df in sCrvOBJ.nodes()]), columns=['date', 'DF', 'zeroRT'])
15 display(dfSG.style.format(fmTSCF))
16 # DF手計算
17 print(f'1yr DF(hc): {1/(1+Rt1y)} :.8f}, ', '# 1yr DF'
18     f'2yr DF(hc): {(1-Rt2y*dfSG.DF[1])/(1+Rt2y)} :.8f}'); '# 2yr DF
19 # 1x2 fwd IX計算
20 fwdIX_1x2 = sCrvOBJ.forwardRate(iDT(dfSG.date[1]), iDT(dfSG.date[2]), dcA365, cmpdSPL)
21 print(f'1x2 fwdIX:{fwdIX_1x2.rate():.5%}, '
22       f'(hc):{dfSG.DF[1]/dfSG.DF[2]-1:.5%}' )

```

| | date | DF | zeroRT |
|---|------------|------------|------------|
| 0 | 2021-08-04 | 1.00000000 | 6.765888% |
| 1 | 2022-08-04 | 0.93457944 | 7.000000% |
| 2 | 2023-08-04 | 0.77743776 | 14.313830% |

図 3.15: シングルカーブのブートストラップ計算例

10: sCrvOBJ : PiecewiseLogLinearDiscount でイールドカーブオブジェクト sCrvOBJ を作成。

13~15: dfSG : date, DF, zeroRT でデータフレームを作成。

- zeroRT を表示させた理由は図 3.1 と比較する為。
- Rt1y=7% は zeroRT も同じ。Rt2y の zeroRT は 14.3% でパーレートの 13% と異なる。
- 異なる理由は“Rt2y は期中にキャッシュフローのあるレート”の為。

17: Tibor の DF 計算 : 式 (1.6) より算出。(数値式は省略)

18: ブートストラップによる D_E 計算 : 式 (3.6) より算出。

$$\overbrace{0.77743776}^{D_E} = \frac{\overbrace{1 - 0.13 \times \overbrace{(0.93457944)}^{\tau_1 \times D_1}}^S}{\underbrace{(1 + 0.13)}_{\tau_2 \times S}}$$

20: forwardRate : カーブオブジェクトの forwardRate メソッドによって、1 年先 2 年満期のフォワード指標金利 (フォワード Tibor) を算出。

- このメソッドの引数は下図 (リファレンスマニュアル) で確認。

◆ forwardRate() [1/3]

```
InterestRate forwardRate (const Date & d1,
                           const Date & d2,
                           const DayCounter & resultDayCounter,
                           Compounding comp,
                           Frequency freq = Annual,
                           bool extrapolate = false) const
```

- 戻り値は interestRate 型であり、値のみは rate() で抽出。
- referenceDate(2021 年 8 月 4 日) より前の日付を第 1 引数に指定する
とエラーとなる点に注意。

22: フォワード指標金利の手計算：式 (2.6) より算出。(数値式は省略)

(確認用スワップオブジェクトの作成)

図 3.15 のカーブを用いた場合、固定クーポン 13% の 2 年スワップは NPV
がゼロになる。その点を確認する為のスワップオブジェクトを作つておく。

ql.MakeVanillaSwap(*tenor, index, fixedRate, forwardStart*)

Optional params:

Nominal

(以下 省略)

```
tenor = ql.Period('5y')
index = ql.Euribor6M()
fixedRate = 0.05
forwardStart = ql.Period("2D")

swap = ql.MakeVanillaSwap(tenor, index, fixedRate, forwardStart, Nominal=100)
swap = ql.MakeVanillaSwap(tenor, index, fixedRate, forwardStart, swapType=ql.VanillaSwap.Payer)
```

上図は Schedule オブジェクト無しで簡便にスワップオブジェクトを作成
できる **MakeVanillaSwap** の QLP ドキュメントである。このコンストラクタ
で確認用スワップオブジェクトを作成したコード例が図 3.16 となる。

初めにコンストラクタの仕様を図 3.16 の 4,5 行目と共に確認する。

必要な引数 : tenor, index, fixedRate, forwardStart の 4 つ。

名前付き引数 : その他は 5 行目のように名前付き引数で指定。

(名前の一覧は QLP ドキュメントを参照)

fixedLegTenor=pdFreqA : 固定レグの利払い頻度を Annual に指定。

nominal=10_000_000 : 想定元本を 1000 万円に指定。

tenor, fixedRate : この 2 つの引数は自明。

- 2、3 行目 swapTNR=pdD('2y'), cpnRT=Rt2y を指定。

index : 2番目の引数はカーブハンドルが与えられた指標金利 IX1y を指定。

(3.4.3節ではこの index をツーカーブで計算された指標金利 twIX1y に修正)

- 図 3.15 の 5 行目で IX1y はカーブハンドル sCrvHDL と共に作成。

forwardStart : 4番目の引数はスワップの起算日をイールドカーブ決済日 (referenceDate, 2021年8月4日) からの日数を期間オブジェクト (Period) で指定。

- スポットスタートは3行目の pD('0d') でゼロ日を指定。

| | | |
|---|---|----|
| 1 | # スワップOBJ | =¥ |
| 2 | swapTNR, cpnRT, fwdStart | |
| 3 | pD('2y'), Rt2y, pD('0d') | |
| 4 | swapOBJ = ql.MakeVanillaSwap(swapTNR, IX1y, cpnRT, fwdStart, | |
| 5 | fixedLegTenor=pdFreqA, nominal=10_000_000) | |
| 6 | # シングルカーブ評価 | |
| 7 | swapOBJ.setPricingEngine(ql.DiscountingSwapEngine(sCrvHDL)) | |
| 8 | print(f'NPV(single curve):{swapOBJ.NPV():.6f}') | |
| 9 | display(mu.swapCashFlow(swapOBJ, sCrvOBJ).style.format(fmtSCF)) | |

| NPV(single curve):0.000000 | | | | | | | | | |
|----------------------------|------------|------------|------------|------------|------|------------|--------|--------------|------------|
| | fixingDate | accruStart | accruEnd | payDate | days | rate | spread | amount | DF |
| 0 | nan | nan | nan | 2021-08-04 | nan | nan% | nan% | nan | 1.00000000 |
| 1 | 2021-08-02 | 2021-08-04 | 2022-08-04 | 2022-08-04 | 365 | 7.000000% | 0.000% | 700,000.00 | 0.93457944 |
| 2 | 2022-08-02 | 2022-08-04 | 2023-08-04 | 2023-08-04 | 365 | 20.212766% | 0.000% | 2,021,276.60 | 0.77743776 |

図 3.16: ブートストラップ計算用スワップデータ

ではコードの他の部分と出力を確認する。

7: DiscountingSwapEngine : 図 3.15 で作成したシングルカーブの sCrvHDL を指定。 (3.4.3 節ではディスカウントカーブ dCrvHDL に修正)

8: swapOBJ.NPV : このプリント行は NPV=0.0 であることを確認する為。

9: swapCashFlow 関数 : myUtil モジュールの swapCashFlow 関数によりキャッシュフロー表を出力。

- rate[2]=20.212766%は図 3.15 の 22 行目の手計算と一致。

(Semiannual のケース)

次に図 3.15 のトイモデルで 1 年 Tibor では無く、6ヶ月 Tibor と 2 年スワップレート (Semiannual) で与えられた場合のブートストラップの計算法の考え方を説明しておく。

まず式(3.5)から次式が成立する。

$$\begin{aligned} & D_{2.0} \times (1 + \tau_{2.0} S) \\ &= 1 - S \times \left(\underbrace{\tau_{0.5} D_{0.5}}_{\substack{\text{(既知)} \\ \text{補間式} \\ \text{が必要}}} + \underbrace{\tau_{1.0} D_{1.0}}_{\substack{\text{(未知数)} \\ \text{補間式} \\ \text{が必要}}} + \underbrace{\tau_{1.5} D_{1.5}}_{\substack{\text{(未知数)} \\ \text{補間式} \\ \text{が必要}}} \right) \end{aligned} \quad (3.7)$$

2年ノードのディスカウントファクター $D_{2.0}$ を求めるには、式(3.7)右辺が示すように、 $D_{2.0}$ より満期の短いディスカウントファクター、 $D_{0.5}, D_{1.0}, D_{1.5}$ がすべて必要となる。^{*12}

$D_{0.5}$ は Tibor レートから計算でき、 $D_{1.0}, D_{1.5}$ は 1.3.2 節で示したように、次の補間式で表すことが出来る。

対数線形補間の式：

$$\begin{aligned} \ln(D_{1.0}) &= [\ln(D_{2.0}) - \ln(D_{0.5})] \times \frac{0.5 \text{ 年} \sim 1.0 \text{ 年の日数}}{0.5 \text{ 年} \sim 2.0 \text{ 年の日数}} + \ln(D_{0.5}) \\ \ln(D_{1.5}) &= [\ln(D_{2.0}) - \ln(D_{0.5})] \times \frac{0.5 \text{ 年} \sim 1.5 \text{ 年の日数}}{0.5 \text{ 年} \sim 2.0 \text{ 年の日数}} + \ln(D_{0.5}) \end{aligned}$$

未知数 3 つに対して、3 つの方程式(式(3.7)と 2 つの補間式)となり、計算可能となる。

ここでは連立方程式の解は示さないが、ブートストラップを行う場合 補間法の指定が必要になることを認識しておこう。

3.4.2 マルチカーブ入門

ではマルチカーブに移ろう。2007 年の世界金融危機以降、担保契約(CSA, Credit Support Annex^{*13})を伴うデリバティブの評価はツーカーブやマルチカーブで行うようになった。^{*14}

ツーカーブ(Two-curve)：

デリバティブからのキャッシュフローを“現在価値に割り引くイールドカーブ”(ディスカウントカーブ)と“将来のキャッシュフローを作るイールドカーブ”(フォワードカーブ)が異なる評価法。

- 担保に付利される金利はオーバーナイトレートが使用される為、ディスカウントカーブは通常 RFR カーブを使用。

^{*12}目的のノードの 1 手前までのディスカウントファクターを用い、目的のノードのディスカウントファクターを算出させる計算法なので、靴紐や“つまみ革”になぞらえ、ブートストラップ法と呼ばれる。

^{*13}ISDA マスター契約を補足する担保付取引の契約書を指す。

^{*14}3.4 節は小川 [3] を参照した。尚 同 [3] で使用した用語はより判り易く直観的な名称に変更した。

- フォワードカーブは多様となるが、スタンダードは標準テナー(円建て、ユーロ建て等は6か月)のIborと固定金利を交換するIborスワップカーブを使用。
- 世界金融危機以前ではこの2つのイールドカーブが同一で、Iborスワップカーブが使用されていた為、ツーカーブに対比させシングルカーブ(Single-curve)として参照される。

マルチカーブ (Multi-curve) :

マルチカーブの場合 次の2つのケースが想定される。

異なる通貨の担保 :

デリバティブとは異なる通貨で担保が提供されるケース。

- このタイプは担保通貨のディスカウントカーブにクロスカレンシーベーシス等で修正を加えたディスカウントカーブを使用。

異なるテナー :

標準テナーとは異なるテナー、例えば1か月Iborや3か月Ibor等のフォワードカーブが使われるケース。

- こちらは標準テナーのフォワードカーブにテナーベーシスで修正を加えたフォワードカーブを使用。

以上をまとめると次のようなイメージ図となる。

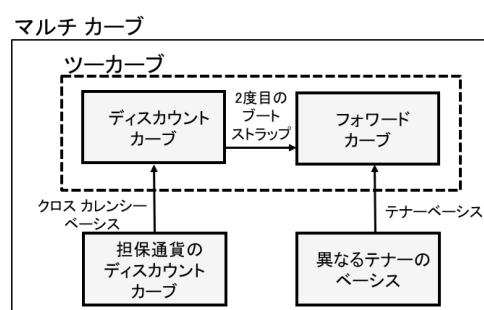


図 3.17: マルチカーブのイメージ

いずれのマルチカーブの計算も図の破線内のツーカーブの計算が要求される為、この3.4節ではツーカーブの計算のみに集中する。

(2度目のブートストラップ)

ツーカーブの場合、式(3.6)で計算する D_E はディスカウントカーブのディスカウントファクターとなり、この D_E を使い、直接フォワードIbor等を計算することは出来ない。

式(2.14)(77ページ)にある $f_{0.5}, \dots, f_E$ を1つづく算出する必要があり、図3.17破線内の**2度目のブートストラップ**(dual bootstrapping)はこのことを意味している。(図3.19で数値例を挙げる)

小川[3]ではこの“2度目のブートストラップ”を省略する次の計算方法が提案されている。

- (1) フォワードカーブとディスカウントカーブのノード E でのベース B_E を算出。(このベースは市場で建値される場合が多い)
- (2) ディスカウントカーブのノード E のアニュイティでベース B_E の現在価値を計算。
- (3) ディスカウントカーブのノード E のディスカウントファクター D_E を B_E の現在価値で調整。

このベースで調整したディスカウントファクター(以下ベース調整DF)を D_E^* で表すと、次式で算出される。

ベース調整DF:

$$D_E^* := D_E - B_E \times \sum_{i=1}^E \tau_i D_i \quad (3.8)$$

シングルカーブのフォワードレート式(2.6)やスワップレート式(2.17)に現れる分子のディスカウントファクターをこのベース調整DFに置き換えることでフォワードカーブ側のフォワードレートやスワップレートが算出可能となる。

フォワードカーブ側 フォワードレート:

$$f_E^* := \frac{D_{E-1}^* - D_E^*}{\tau_E D_E} \quad (3.9)$$

フォワードカーブ側 スワップレート:

$$S_E^* := \frac{\frac{D_S^* - D_E^*}{E}}{\sum_{i=1}^E \tau_i D_i} \quad (3.10)$$

このベース調整DF等の数値例は図3.20で説明する。

3.4.3 ツーカーブでのスワップ評価

(ディスカウントカーブ)

では図3.15のトイモデルに戻り、このイールドカーブはフォワードカーブとしよう。このトイモデルのディスカウントカーブとして、1年指標金利5.0%, 2年スワップ8.0%(Annual)のイールドカーブを追加しよう。

```

1 # Discount curve 金利
2 dCrvHDL = ql.RelinkableYieldTermStructureHandle()
3 dRt1y, dRt2y, dIX1y = ql.Tibor(pdFreqA, dCrvHDL)
4 0.05, 0.08, ql.Tibor(pdFreqA, dCrvHDL)
5 # カーブ構築
6 dHelper = [ql.DepositRateHelper(dRt1y, dIX1y),
7             ql.SwapRateHelper(dRt2y, pd('2y'), calJP, freqA, mFLLW, dcA365, dIX1y)]
8 dCrvOBJ = ql.PiecewiseLogLinearDiscount(Tp2, calJP, dHelper, dcA365)
9 dCrvHDL.linkTo(dCrvOBJ)
10 # DF表示
11 dfDS = pd.DataFrame([(dt.ISO(), df) for dt, df in dCrvOBJ.nodes()], columns=['date', 'DF'])
12 display(dfDS.style.format({'DF': '{:.8f}'}))

      date      DF   1 2022-08-04 0.95238095
0 2021-08-04 1.00000000   2 2023-08-04 0.85537919

```

図 3.18: トイモデルのディスカウントカーブ

図 3.18 の 8 行目でカーブオブジェクト dCrvOBJ を作成した。このコードに関しては本来シングルカーブのブートストラップであり、図 3.15 と同じであるが、次の点を確認しよう。

- 3 行目の変数名は discount の d が最初の一文字となり、2 文字目以下は図 3.15 に同じ。(DF やフォワードレートの手計算は省略)

(フォワードカーブ)

フォワードカーブの金利は図 3.15 シングルカーブと同じとする。フォワードカーブのブートストラップは図 3.17 の説明のようにディスカウントカーブを使って、フォワードカーブと一致するフォワードレートを算出させる。

```

1 # forward curve
2 fCrvHDL = ql.RelinkableYieldTermStructureHandle()
3 fIX1y = ql.Tibor(pdFreqA, fCrvHDL)
4 # カーブ構築
5 fHelper = [ql.DepositRateHelper(Rt1y, fIX1y),
6             ql.SwapRateHelper(Rt2y, pd('2y'), calJP, freqA,
7                                 mFLLW, dcA365, fIX1y, mu.sqHDL(0), pd('0d'), dCrvHDL)]
8 fCrvOBJ = ql.PiecewiseLogLinearDiscount(Tp2, calJP, fHelper, dcA365)
9 fCrvHDL.linkTo(fCrvOBJ)
10 # DF, fwdRT表示
11 dfFW = pd.DataFrame(
12     [(dt.ISO(), df) for dt, df in fCrvOBJ.nodes()], columns=['date', 'DF'])
13 display(dfFW.style.format({'DF': '{:.10f}'}))
14 fwdIX_1x2 = fCrvOBJ.forwardRate(iDT(dfFW.date[1]), iDT(dfFW.date[2]), dcA365, cmpdSPL)
15 print(f'1x2 fwdIX:{fwdIX_1x2.rate():.5%}')

      date      DF   1 2022-08-04 0.9345794393
0 2021-08-04 1.0000000000   2 2023-08-04 0.7808959050   1x2 fwdIX:19.68041%

```

図 3.19: トイモデルのフォワードカーブ

この 2 度目のブートストラップ処理を QuantLib で行うコードが図 3.19 となる。図 3.15 シングルカーブのコードとほぼ同じであるが、次の異なる部分

を確認しよう。

3, 5: fix1y : 変動金利指標の名称に forward の f を追加。

6, 7: SwapRateHelper : このコンストラクタを説明した図 2.4(53 ページ)を参照し、次の 4 つの引数を確認。

7 番目引数: iborIndex : フォワードカーブ用に作成した変動金利指
数 fix1y を指定。

8,9,10 番引数: mu.sqHDL(0), pD('0d'), dCrvHDL :

10 番目の引数 dCrvHDL をディスカウントカーブとして設定する
為に 8,9 番の引数を挿入している。

8 番目: spread : 本来 iborIndex に対するスプレッドを設定。この
例は mu.sqHDL(0) によって、spread=0 とした。

9 番目: fwdStart : pD('0d') 指定により、スポットスタートを
指定。

10 番目: discountingCurve : 図 3.18 で作成した dCrvHDL ハン
ドルを指定。

14, 15: forwardRate : 図 3.15 の 20,21 行目とこの 14,15 行目ではオブジェ
クトが sCrvOBJ と fCrvOBJ で異なっている。

- 算出された 1×2 fwdIX:19.6804% はシングルカーブの 20.2127% と
異なる点に注意。
- 前者の 19.6804% がツーカーブで必要となるフォワード Tibor であ
り、“2 度目のブートストラップ”によって算出された値。
- このフォワード Tibor 19.6804% は出力されたデータフレーム dfFW
の DF を使い、 $0.934579 / 0.7808959 - 1$ により計算可能。(ただし、
dfFW の DF から 2 年スワップレート 13.0% を算出出来ない点に注意)

(2 度目のブートストラップの手計算)

図 3.20 では式 (3.8)、(3.9) を使い、フォワード Tibor 19.6804% と 2 年ス
ワップレート 13.0% を手計算した。簡単にコードを見ておく。

2: bs1y, bs2y : フォワードカーブとディスカウントカーブの 2 つのノード
の金利の差 (ベース) を bs1y, bs2y に設定。

3, 4: ベーシス調整 DF: a1DF, a2DF : 式 (3.8) のベーシス調整 DF を計
算。(a は adjusted の略)

```

1 # ベース調整DF
2 bs1y, bs2y = Rt1y-dRt1y, Rt2y-dRt2y
3 a1DF = dfDS.DF[1] - bs1y*dfDS.DF[1]
4 a2DF = dfDS.DF[2] - bs2y*(dfDS.DF[1]+dfDS.DF[2])
5 print(f'ベース調整DF 1y:{a1DF:.8f}, , f' 2y:{a2DF:.8f}')
6 # フォワードIbor, スワップレート
7 print(f'1x2 fwdIX(hc):{(a1DF - a2DF)/dfDS.DF[2]:.5%}, ,
8 f' 2y swap(hc):{(1 - a2DF)/(dfDS.DF[1]+dfDS.DF[2]):.5%}' )

```

ベース調整DF 1y:0.93333333, 2y:0.76499118
1x2 fwdIX(hc):19.68041%, 2y swap(hc):13.00000%

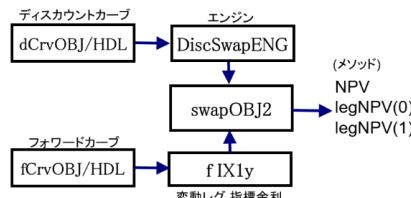
図 3.20: ベース調整 DF によるフォワード Tibor 等の計算例

7: **1×2 fwdIX(hc)** : 式 (3.9) を使い、1年先フォワード Tibor 19.6804%を算出。

8: **フォワードカーブの2年スワップレート** : 式 (3.10) によって、フォワードカーブの2年スワップレート 13.0%を算出。

(ツーカーブ・スワップ評価)

ディスカウントカーブ(図 3.18)とフォワードカーブ(図 3.19)の構築が終わり、ようやくスワップの評価が可能となる。



```

1 # fIX1y指数のスワップOBJとエンジンの設定
2 swapOBJ2 = ql.MakeVanillaSwap(swapTNR, fIX1y, cpnRT, fwdStart,
3                                fixedLegTenor=pdFreqA, nominal=10_000_000)
4 swapOBJ2.setPricingEngine(ql.DiscountingSwapEngine(dCrvHDL))
5 # スワップ評価
6 print(f'NPV(Two curve):{swapOBJ2.NPV():,.2f}, ,
7       f'FixLeg:{swapOBJ2.legNPV(0):,.2f}, FltLeg:{swapOBJ2.legNPV(1):,.2f}')
8 # キャッシュフロー表
9 display(mu.swapCashFlow(swapOBJ2, fCrvOBJ, 1).style.format(fmtSCF)) # float
10 display(mu.swapCashFlow(swapOBJ2, dCrvOBJ, 0).style.format(fmtSCF)) # fix

```

NPV(Two curve):0.00, FixLeg:-2,350,088.18, FltLeg:2,350,088.18

| | fixingDate | accruStart | accruEnd | payDate | days | rate | spread | amount | DF |
|---|------------|------------|------------|------------|------|------------|--------|--------------|------------|
| 0 | nan | nan | nan | 2021-08-04 | nan | nan% | nan% | nan | 1.00000000 |
| 1 | 2021-08-02 | 2021-08-04 | 2022-08-04 | 2022-08-04 | 365 | 7.000000% | 0.000% | 700,000.00 | 0.93457944 |
| 2 | 2022-08-02 | 2022-08-04 | 2023-08-04 | 2023-08-04 | 365 | 19.680412% | 0.000% | 1,968,041.24 | 0.78089590 |

| | nominal | accruStart | accruEnd | payDate | days | rate | amount | DF |
|---|---------------|------------|------------|------------|------|------------|--------------|------------|
| 0 | nan | nan | nan | 2021-08-04 | nan | nan% | nan | 1.00000000 |
| 1 | 10,000,000.00 | 2021-08-04 | 2022-08-04 | 2022-08-04 | 365 | 13.000000% | 1,300,000.00 | 0.95238095 |
| 2 | 10,000,000.00 | 2022-08-04 | 2023-08-04 | 2023-08-04 | 365 | 13.000000% | 1,300,000.00 | 0.85537919 |

図 3.21: ツーカーブでのスワップ評価

ツーカーブでスワップを評価する場合、次の 2 点が重要である。

- フォワードカーブで設定した指標金利オブジェクトでスワップオブジェクトを作り直す。(図 3.19 で計算したフォワード Tibor をスワップオブジェクトで計算出来るようにする為)
- スワップのエンジンはディスカウントカーブを指定。(キャッシュフローのディスカウントは当然ディスカウントカーブで行う)

この 2 点に注意して、図 3.21 のコードと出力を確認する。

- 2: **fIX1y** : MakeVanillaSwap の 2 番目の引数を図 3.19 で作成した fIX1y を指定し、swapOBJ2 を新たに作成。
- 4: **dCrvHDL** : スワップの評価エンジンとして、図 3.18 で作成した dCrvHDL ハンドルを設定。
- 6: **swapOBJ2.NPV()** : スワップ NPV は 0.00 であり、固定レグと変動レグの時価は釣り合っている。
- 7: **legNPV(0), legNPV(1)** : 固定レグ(0) と変動レグ(1) の時価 2,350,088.18 円をプリント。
 - 10 行目で出力された固定レグキャッシュフロー表で amount 列 × DF 列の合計がこの金額となる。(各自確認)
- 9, 10: **swapCashFlow 関数** : キャッシュフロー表を表示させたが、9 行目(変動レグ) と 10 行目(固定レグ) の swapCashFlow 関数の第 2 引数を fCrvOBJ と dCrvOBJ としている点に注意。
 - 変動レグ(9 行目) のキャッシュフロー表の DF 列は rate 列から逆算された DF であり、現在価値を算出する DF ではない点に注意。(図 3.19 で説明したようにスワップレートの算出も出来ない)
 - 変動レグ(9 行目) の第 2 引数を dCrvOBJ とすると、rate 列に表示されるフォワード Tibor が dCrvOBJ から算出する数値となる点に注意。

図 3.21 のオブジェクト図とシングルカーブのスワップのオブジェクト図(57 ページ図 2.8) を比較して理解しよう。

